© 2021 Meghan Kelley

VARIATIONS OF ONLINE BIPARTITE MATCHING

BY

MEGHAN KELLEY

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Computer Science
in the Graduate College of the
University of Illinois Urbana-Champaign, 2021

Urbana, Illinois

Adviser:

Professor Sheldon H. Jacobson

ABSTRACT

The Online Bipartite Matching Problem is a well studied problem in theoretical computer science that models several real world applications including online investment, kidney transplantation, aviation security passenger screening, and enhanced Ebola entry screening. However, the original version of the problem is too simplistic to cover many real world applications. Therefore it is common to consider variations of the problem that more closely model the target application.

This thesis considers two variations of the problem motivated by aviation security passenger screening. The first, known as the online total bipartite matching problem, is a variation in which jobs must be assigned to some worker regardless of whether or not it is adjacent to an available worker. Tight upper and lower bounds are given for the general version of this problem, along with 1-competitive algorithm for a special case of the problem.

The second variation begins with the well known Stochastic Sequential Assignment Problem, which is a variation of the Online Bipartite Matching problem in which edge weights are calculated as the product of a job value and worker value. It extends this to the Reusable Sequential Stochastic Assignment Problem, in which workers can be reused after they finish processing a job. We consider both the stochastic and random arrival model and provide algorithms with constant approximation ratios when job lengths are constant.

ACKNOWLEDGMENTS

I would first like to thank my advisor Dr. Sheldon Jacobson for his support through the research and writing process. Through his guidance and expertise I have learned a great deal about choosing problems, exploring different possibilities, and communicating research.

In addition, I would like to thank my parents for their love and guidance. They made sure to teach me the value of discipline and hard work, which has helped me to get to this point. Even today, their constant advice and support is invaluable.

This research was supported in part by the Air Force Office of Scientific Research (FA9550-19-1-0106). Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the United States Air Force, Department of Defense, or the United States Government.

TABLE OF CONTENTS

CHAPTER 1: INTRODUCTION

The *Online Bipartite Matching Problem* (OBMP) is defined by a finite (known) number of workers and a finite number of sequentially arriving jobs. Every job-worker pair has an associated reward that is earned when that job is assigned to that worker. The objective is to assign each job to a worker so as to maximize the total reward.

The model of bipartite online matching embraces a broad area of research and application scenarios, such as online investment [1], kidney transplantation [2], aviation security passenger screening [3], and enhanced Ebola entry screening [4]. The problem is first defined in [5], which gives both a 1/2-competitive deterministic algorithm and a $1 - 1/e$-competitive randomized algorithm and shows that these are optimal. However, many of these applications are not directly modeled by OBMP as stated, and so these results cannot often not be directly applied to real world problems. Therefore, it is important to consider variations that bring us closer to more accurately modeling real world situations. Several variations of Online Bipartite Matching have already been studied. For example, [6] shows that it is impossible to improve upon the 1/2-competitive deterministic and $1 - 1/e$-competitive randomized algorithm when jobs are revealed in blocks of size at most $o(n)$. [7] considers a version of OBMP where jobs are drawn from a known independent and identically distributed (*iid*) distribution, and gives an algorithm with competitive ratio better than $1-1/e$ in this model. Several others ([8, 9, 10, 11]) have explored this variation of the problem further. [12] and [13] introduce different types of weighted models for this problem. Newer results such as [14], [15], [16], and [17] have considered other variations of online bipartite matching. For a more complete survey of work on similar types of matching problems, see [18].

We will examine two variations of the problem motivated by aviation security passenger screening. Aviation security passenger screening has previously been modeled by a variation of OBMP known as the Sequential Stochastic Assignment Problem (SSAP) [19]. In SSAP, workers are associated with known values, and incoming jobs have values drawn from an *iid* distribution. The product of these values for a given job-worker pair is the weight of the associated edge.

Similarly to the standard OBMP problem, several SSAP variations have been previously studied. [20] provides optimal policies for SSAP with random arrival times and discounted rewards under different arrival distributions and various discount functions. [21] consider SSAP with a random number of arriving jobs. They propose optimal policies using an auxiliary SSAP instance for two cases: the total number of arriving jobs is unknown but

finite; and the total number of arriving jobs can be infinite. [22] considers SSAP with the distributions of two successive job values governed by a Markov chain. [23] relaxes this Markov chain assumption and considers SSAP with the distributions of two successive job values governed by a partially observable Markov chain. [24] considers SSAP where the distributions of successive job values are not necessarily independent. [25] consider an SSAP with binary eligibility vectors. Another SSAP extension is the *multi-objective* SSAP, where the objective is to maximize several objective functions simultaneously. [26] analyze a bi-objective multi-criteria Secretary Problem, a special class of SSAP, and propose an algorithm for generating the complete set of Pareto optimal policies by combining the optimal policy of SSAP and the weighted sum method.

In the context of aviation screening, workers model screening checkpoints that have a known security value, and jobs model passengers that have an estimated risk value that is revealed upon arrival. [27] uses SSAP to study the *Sequential Stochastic Security Design Problem* (SSSDP), which maximizes the system security over a finite-time horizon based on passenger risks and security parameters. [28] introduces the *Sequential Stochastic Passenger Screening Problem* (SSPSP), which optimally assigns (in real-time) passengers to security resources. [29] uses a discrete time difference equation to model passenger assignment in real-time. [30] introduces the multi-stage sequential passenger screening problem (MSPSP) for passenger screening, with real-time updates to perceived passenger risk.

The first challenge in using SSAP to model aviation security passenger screening systems that we will consider is how performance is measured. In particular, the objective function captures the expected total reward, which is an overall system performance measure. It does not measure the impact on individual passenger screening. In many situations, it is important to ensure that the security screening procedure assigned to *each* passenger is consistent with their perceived risk. These situations can be modeled by an unweighted OBMP instance in which there is an edge between a passenger and a screening procedure if the screening procedure meets the security threshold needed for the passenger's risk level. However, in traditional OBMP, jobs can be discarded and not assigned to any worker. In an aviation security setting, this is unacceptable, since it translates to allowing a passenger to pass through security without any screening. To address this problem we introduce the Online Total Bipartite Matching problem (OTBMP), in which each job must be assigned to a worker, even if there are no available edges.

We also separately address a second problem that occurs when trying to use SSAP to model passenger screening is that of resuability. In particular, in the case of passenger screening checkpoint can generally be reused once it has finished processing a passenger. However, current SSAP literature generally only considers the case where workers, once

assigned to a job, are no longer available for the remainder of the input stream. To address problem, we introduce the Reusable Sequential Stochastic Assignment Problem (RSSAP), in which workers can be reused after they have finished processing a job.

Our results are organized in the following manner: Chapter 2 formally introduces the Online Total Bipartite Matching problem. It then provides tight lower and upper bounds for both the most general versions of the problem as well as special cases that more closely model aviation security passenger screening. Chapter 3 introduces the Reusable Sequential Stochastic Assignment Problem, and provides constant approximation ratios both when arrivals are IID and when they are adversarially chosen but have a random arrival order.

CHAPTER 2: ONLINE TOTAL BIPARTITE MATCHING PROBLEM

This chapter introduces a variation of the unweighted OBMP, titled the Online Total Bipartite Matching Problem (OTBMP). Recall that in standard OBMP, jobs can be discarded rather than assigned to a worker. In OTBMP, each job must be assigned to a worker, even if there are no available edges. This chapter begins by formally describing OTBMP in Section 2.1. Section 2.2 gives assignment policies for the most general version of OTBMP and shows that no randomized policy has constant competitive ratio. However, it often makes sense to consider restricted instances that allow us to do better than the given lower bounds. In particular, it is often the case that workers can be ordered by their ability, and any worker that can complete a more difficult job can also complete a less difficult one. We refer to this property as *order preserving*. Section 2.3 considers the assignment policy which assigns each arriving job to the least capable worker that can successfully complete that job and shows that this assignment policy is optimal for OTBMP instances with the order preserving property. In addition, it identifies the most general condition under which the given policy is guaranteed to be optimal.

2.1 OTBMP FORMULATION

In the online bipartite matching problem, we start with a known worker set, and a known set from which jobs can be drawn. Jobs arrive one at a time, and upon arrival, must be assigned to some worker or discarded. The goal is to maximize the total weight of the assignments.

In this chapter, we introduce the *Online Total Bipartite Matching Problem* (or OTBMP), in which vertices cannot be discarded. Formally, consider an $(\mathscr{X}, P)$-partite graph (known as the *underlying graph*) with both $\mathscr{X}$ (the universe of jobs) and $P = \{p_1, p_2, \ldots, p_n\}$ (the set of workers) known in advance. Edges are defined by a weight function $w : \mathscr{X} \times P \to \mathbb{R}$, which is also known in advance. Jobs arrive sequentially in the order $X = (x_1, x_2, \ldots, x_n)$ where $x_i \in \mathscr{X}$ for all i; we say that round i begins when job x_i arrives and ends when job x_{i+1} arrives. The job x_i must be assigned to an available worker prior to the next job's arrival. In particular, this means an assignment must be made before the rest of the sequence is known. Once a job is assigned to a worker, its assignment cannot be changed. In addition, only one job can be assigned to each worker. The goal is to assign workers to jobs so as to maximize

$$r(\sigma, P) = \sum_{i=1}^{n} w(x_i, \sigma(x_i)) \tag{2.1}$$

where $\sigma(x_i)$ is the worker to which job x_i is assigned. This chapter will only consider the unweighted version of the problem, that is, we require that $w(x_i, p_j) \in \{0, 1\}$ for all $x_i \in \mathscr{X}$ and $p_j \in P$. We say x_i is adjacent to p_j if and only if $w(x_i, p_j) = 1$.

In order to discuss the quality of algorithms that solve this problem, we define the competitive ratio.

Definition 2.1. Consider an online maximization problem $\mathscr{P}$. Let ALG be a randomized algorithm that attempts to solve $\mathscr{P}$. Assume there exists an adversary that is allowed to construct an instance I of $\mathscr{P}$ with knowledge of ALG, but without knowledge of the randomness it will use (this is known as an oblivious adversary). We say that ALG is *α-competitive against an oblivious adversary* if for any instance I constructed by such an adversary,

$$\mathbb{E}[\text{ALG}(I)] \geq \alpha \cdot \text{OPT}(I) + \beta, \tag{2.2}$$

where $\text{OPT}(I)$ is the offline optimal solution for instance I and β is a constant independent of I.

2.2 GENERAL GRAPHS

Theorem 2.1 shows that the best policy for OTBMP cannot be better than $1/n$-competitive.

Theorem 2.1. No policy for OTBMP is better than $1/n$-competitive against an oblivious adversary.

Proof. Consider an instance where $P = \{p_1, p_2, \ldots, p_n\}$ and $\mathscr{X} = \{x^{(0)}, x^{(1)}, \ldots, x^{(n)}\}$. Define w so that $w(x^{(i)}, p_j) = 1$ if and only if $i = j$. Let π be an arbitrary randomized assignment policy. The adversary will first send $x_1, x_2, \ldots, x_{n-1} = x^{(0)}$ (i.e., all edges adjacent to these jobs have weight 0). Then, the adversary will send a job $x_n = x^{(j)}$ where p_j is the worker that is least likely to be available after the previous $n-1$ assignments have been made. Note that the adversary can compute this with knowledge of π. Since p_j is the worker least likely to be available in round n, p_j is available with probability at most $\frac{1}{n}$. Therefore the expected reward for π is at most $\frac{1}{n}$, while the optimal offline assignment has a reward of 1. Hence for any assignment policy π, an oblivious adversary can construct a family of instances for which $\mathbb{E}[\pi(I)] \leq \frac{1}{n}\text{OPT}(I)$. QED.

The algorithm that matches every incoming job to a random worker is $1/n$-competitive; therefore this bound is tight.

Since we know no policy for OTBMP has constant competitive ratio, it makes sense to consider instances with reasonable restrictions. The proof of Theorem 2.1 relies on the fact

that $\mathscr{X}$ includes a job that is not adjacent to any worker. However, in many contexts, the universe of jobs will not contain such jobs. We will show that even in these cases, there is no constant competitive algorithm for OTBMP, although the competitive ratio does improve to $\Theta(1/\sqrt{n})$.

Theorem 2.2. No policy for OTBMP is better than $O(1/\sqrt{n})$-competitive against an oblivious adversary, even when for every job $x \in \mathscr{X}$ there exists some worker $p_j \in P$ such that $w(x, p_j) = 1$.

Proof. Consider an instance where $P = \{p_1, p_2, \ldots, p_n\}$ and $\mathscr{X} = \{x^{(1)}, x^{(2)}, \ldots, x^{(n)}\}$. Define w so that $w(x^{(i)}, p_j) = 1$ if and only if $i = j$. Let π be an arbitrary randomized assignment policy. Assume there are n workers. The adversary will first send $x_1, x_2, \ldots, x_{n-\sqrt{n}} = x^{(1)}$. The remaining $\sqrt{n}$ jobs will have a perfect matching with the $\sqrt{n}$ workers least likely to be available after the first $n-\sqrt{n}$ jobs have been assigned (and no other edges). In particular, if the set of jobs least like to be available is $\{p_{j_1}, p_{j_2}, \ldots, p_{j_{\sqrt{n}}}\}$, then we can set $x_{n-\sqrt{n}+i} = x^{(j_i)}$ for $1 \leq i \leq \sqrt{n}$. As before, the adversary can compute this with knowledge of π.

The first $n - \sqrt{n}$ jobs can earn a reward of at most 1 between them. Each worker besides p_1 has a probability of at most $\frac{\sqrt{n}}{n-1}$ of being available after the first $n - \sqrt{n}$ jobs have been assigned. Therefore the total expected reward,

$$\mathbb{E}[\pi(I)] \leq 1 + \frac{(\sqrt{n})(\sqrt{n})}{n-1} = 1 + \frac{n}{n-1} \leq 3. \tag{2.3}$$

In addition, $\mathrm{OPT}(I) = \sqrt{n} + 1$, since the first job can be matched to p_1 and the last $\sqrt{n}$ jobs can be matched according to the perfect matching in an offline setting. Therefore

$$\mathbb{E}[\pi(I)] \leq 3 = \frac{3}{\sqrt{n}+1}\mathrm{OPT}(I), \tag{2.4}$$

and no policy for OTBMP is better than $O(1/\sqrt{n})$-competitive. QED.

The rest of this section will be devoted to showing that Theorem 2.2 is tight up to constant factors by giving an $\Omega(1/\sqrt{n})$-competitive algorithm for OTBMP in this setting. We use an algorithm that assigns a job to an arbitrary available adjacent worker if such a worker is available. Otherwise we assign the job to a random available worker. We will refer to this algorithm as "Greedy-Random", or "GR". The algorithm is described formally by Algorithm 2.1

Theorem 2.3. GR is $\Omega(1/\sqrt{n})$-competitive when every job is adjacent to at least one worker.

Algorithm 2.1 GR algorithm for the OTBMP problem

Assume we have a known worker set $P = \{p_1, p_2, \ldots, p_n\}$, known job universe $\mathscr{X}$, and known weight function $w : \mathscr{X} \times P \to \{0, 1\}$
$P_1 \leftarrow P$
for all jobs x_i arriving **do**
    $\sigma(x_i) \leftarrow$ None
    for all workers p_k such that $w(x_i, p_k) = 1$ **do**
        if $\sigma(x_j) \neq p_k$ for all $j < i$ (that is, p_k is available) **then**
            $\sigma(x_i) \leftarrow p_k$
            Break
        end if
    end for
    if $\sigma(x_i)$ is still None **then**
        $\sigma(x_i) \leftarrow$ a random worker from P_i
    end if
    $P_{i+1} \leftarrow P_i \setminus \{\sigma(x_i)\}$
end for

Proof. Consider an arbitrary instance of OTBMP where w is defined so that for every $x \in \mathscr{X}$, there is some $p_j \in P$ such that $w(x, p_j) = 1$. Let GR be the random variable corresponding to the reward earned by GR on the instance, and let OPT be the reward earned by the offline OPT. Let $O(x_i)$ be the worker to which job x_i is matched in the offline OPT. Let $M \equiv \{x_i \mid w(x_i, O(x_i)) = 1\}$, and let M' be initialized to the the $\lfloor \text{OPT}/2 \rfloor$ vertices from M that arrive first.

We know that job x_1 will always earn a reward of 1 using GR, since it must be adjacent to some worker, and that worker must be available as no other jobs have been assigned. If x_1 is in M', remove it. In addition, if x_1 is assigned to worker p_k and there is a job $x_j \in M'$ such that $O(x_j) = p_k$, remove x_j from M'

Consider each job $x_i \in M'$ in the order they arrive. When job x_i arrives, if it assigned to worker p_k and there is a job $x_j \in M'$ such that $O(x_j) = p_k$, remove x_j from M'. In this way, we guarantee that $O(x_i)$ is available unless it was randomly chosen by a job with no available adjacent workers. Then the probability that $O(x_i)$ is available is at least $\frac{\lceil \text{OPT}/2 \rceil}{n}$. This is because there are at least $\lceil \text{OPT}/2 \rceil$ workers that are unassigned when x_i arrives (corresponding to the second half of M that is never considered in the analysis), and there are at most n vertices that this set of $\lceil \text{OPT}/2 \rceil$ could be chosen from. Therefore with probability at least $\frac{\lceil \text{OPT}/2 \rceil}{n}$, job x_i has at least one available adjacent vertex that it can be assigned to and therefore earns some reward.

We recall that M' has size at least $\lfloor \text{OPT}/2 \rfloor - 2$ after processing job x_1, and at most half

of the remainder of M' is removed. Therefore we can write

$$\mathbb{E}[\mathrm{GR}] \geq 1 + \left(\frac{\lceil \mathrm{OPT}/2 \rceil}{n}\right)\left(\frac{1}{2}\right)(\lfloor \mathrm{OPT}/2 \rfloor - 2) \tag{2.5}$$

$$\geq 1 + \left(\frac{\mathrm{OPT}/2}{n}\right)\left(\frac{1}{2}\right)(\mathrm{OPT}/2 - 3) \tag{2.6}$$

$$= 1 + \frac{\mathrm{OPT}^2 - 6\mathrm{OPT}}{8n}. \tag{2.7}$$

If $\mathrm{OPT} \geq \sqrt{8n}$, we can write

$$\mathbb{E}[\mathrm{GR}] \geq 1 + \frac{\mathrm{OPT}^2 - 6\mathrm{OPT}}{8n} \tag{2.8}$$

$$\geq \frac{\sqrt{8n}\mathrm{OPT} - 6\sqrt{8n}}{8n} \tag{2.9}$$

$$= \frac{\mathrm{OPT} - 6}{\sqrt{8n}} \tag{2.10}$$

$$\geq \frac{\mathrm{OPT}}{\sqrt{8n}} - 2.5, \tag{2.11}$$

where the final inequality assumes $n \geq 1$, and therefore $6/\sqrt{8n} \leq 6/\sqrt{8} \leq 2.5$. Therefore in this case, GR is $1/(\sqrt{8n})$-competitive.

If $\mathrm{OPT} \leq \sqrt{8n}$, we can use the fact that GR must earn a reward for x_1 at least to write

$$\mathbb{E}[\mathrm{GR}] \geq 1 \geq \frac{\mathrm{OPT}}{\sqrt{8n}}, \tag{2.12}$$

and therefore in this case, GR is also $1/(\sqrt{8n})$-competitive. QED.

The fact that $\Theta(1/\sqrt{n})$-competitive is optimal leads to the question of whether this result can be improved for any useful restricted classes of graphs. This will be considered in section 2.3.

2.3 ORDER PRESERVING GRAPHS

We will consider instances of OTBMP in which the underlying graph is order preserving, as defined in Definition 2.2.

Definition 2.2. An unweighted $(\mathscr{X}, P)$-partite with edge weights w is *order preserving* if there is some ordering of the set P, $(p_1, p_2, \ldots, p_n)$, such that for any vertex $x \in \mathscr{X}$, if

$w(x, p_i) = 1$, then $w(x, p_j) = 1$ for all $j \leq i$. Any ordering of workers that satisfies this condition is known as an *order preserving sequence*.

Intuitively, this means that there is some ranking of the workers such that highly ranked workers can complete a larger subset of the jobs than workers with lower rank.

On this type of problem, an intuitive algorithm is to assign jobs to the least valuable worker that they are adjacent to, leaving more valuable workers open for future jobs. Formally, consider an order preserving $(\mathscr{X}, P)$-partite graph with order preserving sequence $(p_1, p_2, \ldots, p_n)$. Define the policy π_{HIF}, termed the Highest Index First (HIF) policy, as follows: when job x_i arrives, assign it to worker p_{j_i} where $j_i \equiv \max\{k \in \{1, 2, \ldots, n\} \ : \ p_k \in P_i \text{ and } w(x_i, p_k) = 1\}$. If no such j_i exists, then $j_i \equiv \max\{k \in \{1, 2, \ldots, n\} \ : \ p_k \in P_i\}$.

Theorem 2.4 proves that the HIF policy is optimal for an order-preserving instance of OTBMP. Note that this result was also proven by [31]. A proof is included here for completeness.

Theorem 2.4. Consider an order preserving $(\mathscr{X}, P)$-partite graph with order preserving sequence $(p_1, p_2, \ldots, p_n)$. Let P_i be the set of workers that are available when job x_i arrives. Policy π_{HIF} produces an optimal assignment for this OTBMP instance.

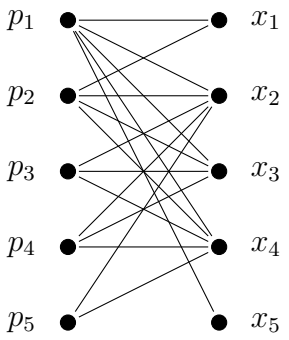
Proof. The proof proceeds by induction on n. In the base case, when $n = 1$, there is only one possible assignment. Therefore any policy will return the optimal solution.

Assume that the HIF policy gives optimal results when the number of workers is at most $n - 1$. Suppose that policy π_{HIF} leads to jobs $x_1, x_2, \ldots, x_n$ begin assigned to workers $p_{h_1}, p_{h_2}, \ldots, p_{h_n}$ respectively, resulting in total reward r_{HIF}.

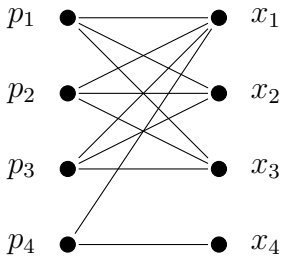
Suppose that there exists an alternate assignment policy, π_{ALT}, which leads to jobs $x_1, x_2, \ldots, x_n$ begin assigned to workers $p_{a_1}, p_{a_2}, \ldots, p_{a_n}$, respectively, resulting in total reward r_{ALT}. Let k denote the index of the first job assigned to a worker that is different between the two policies (i.e., $p_{h_i} = p_{a_i}$ for $i = 1, 2, ..., k-1$, and $p_{h_k} \neq p_{a_k}$).

If $a_k < h_k$, then either both policies earned a reward for this assignment, or both did not (if x_k was adjacent to p_{a_k} but not p_{h_k}, then the HIF policy would not have assigned job x_k to worker p_{h_k} since there was at least one available adjacent job). Therefore the two instances have the same reward at the end of round k. In addition the remaining instance under π_{HIF} has an optimal reward that is at least that of the remaining instance under π_{ALT} (the only difference between the two is that π_{HIF} has p_{a_k} remaining instead of p_{h_k}; but any job adjacent to p_{h_k} is also adjacent to p_{a_k}). Since the HIF policy is optimal on the remaining instance, $r_{\text{HIF}} \geq r_{\text{ALT}}$.

If $a_k > h_k$, then x_k must be adjacent to p_{h_k} and not p_{a_k} (since otherwise the HIF policy would prefer the assignment to p_{a_k}). Therefore π_{HIF} has earned exactly one more reward than



(a) Graph for Example 2.1.



(b) Graph for Example 2.2.

Figure 2.1: OTBMP Examples

π_{ALT} at the end of round k. In addition, the remaining instance under π_{ALT} has an optimal reward that is at most one more than that of the remaining instance under π_{HIF}. Since the HIF policy is optimal on the remaining instance, $r_{\text{HIF}} \geq r_{\text{ALT}}$. QED.

We provide two examples to illustrate the necessity of the order preserving policy in establishing the optimality of the HIF policy. Example 2.1 demonstrates that the policy is optimal, while Example 2.2 demonstrates that without the order preserving property, the policy may not be optimal.

Example 2.1. Consider the OTBMP instance with underlying graph shown in Figure 2.1a. Note that $(p_1, p_2, p_3, p_4, p_5)$ is an order preserving sequence. Assume jobs arrive in the order $(x_1, x_2, x_3, x_4, x_5)$. The HIF policy results in jobs x_1, x_2, x_3, x_4, x_5 being assigned to workers p_2, p_5, p_4, p_3, p_1 respectively. This results in an objective value of 5, and hence, the HIF policy is optimal. Note that this solution is not unique. For example, a solution that assigns these jobs to workers p_2, p_5, p_3, p_4, p_1 also results in an objective value of 5.

Example 2.2. Consider the OTBMP instance with underlying graph shown in Figure 2.1b. Note that the graph is not order preserving, since x_1 is adjacent to p_1, p_2, and p_3 while x_4 is adjacent only to p_4. Assume jobs arrive in the order (x_1, x_2, x_3, x_4). While using the given ordering, the HIF policy assigns jobs x_1, x_2, x_3, x_4 to workers p_4, p_3, p_2, p_1 respectively. This results in an objective value of 3. However, a policy that assigns these jobs to workers p_1, p_2, p_3, p_4 respectively is optimal, with objective value 4.

Example 2.2 demonstrates that the HIF policy is not necessarily optimal on graphs that are not order preserving. However the order preserving property does not accurately model all applications. This leads to the following question: Is order preserving the most general

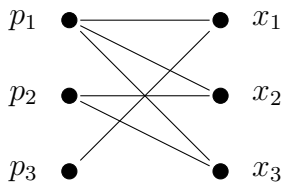


Figure 2.2: Graph for Example 2.3.

condition under which the HIF policy is optimal? Example 2.3 provides an instance of OTBMP in which the underlying graph is not order preserving, yet the HIF policy is optimal.

Example 2.3. Consider an instance of OTBMP with underlying graph shown in Figure 2.2. Note that the graph is not order preserving since x_1 is adjacent to p_3 but not p_2 while x_2 and x_3 are adjacent to p_2 but not p_3. Assume jobs arrive in the order (x_1, x_2, x_3), and that the workers are ordered (p_1, p_2, p_3). When the HIF policy is applied with the given worker ordering, it assigns jobs x_1, x_2, x_3 to workers p_3, p_2, p_1 respectively. This results in objective value of 3. Hence the HIF policy is optimal on this instance.

Example 2.3 suggests that the HIF policy remains optimal even when less restrictive conditions are placed on the underlying graph. This leads us to suggest a second set of graphs on which the HIF policy is optimal.

Let G be an $(\mathscr{X}, P)$-partite graph such that $\mathscr{X} = (x^{(1)}, x^{(2)}, \ldots, x^{(m)})$. Let $S_i \subseteq P$ be the set of workers adjacent to job $x^{(i)}$, and let $\mathscr{S} \equiv \{S_1, S_2, \ldots, S_m\}$.

Intuitively, when $|\mathscr{X}|$ is small, an adversary does not have many different options to choose from. This makes it difficult to construct an instance on which HIF does not perform well. In particular, when $|\mathscr{X}| = 2$, we can guarantee that there is some ordering of workers under which HIF will be optimal. That ordering is characterized by Definition 2.3.

Definition 2.3. If $|\mathscr{X}| = 2$, then let S and T be the two distinct sets in $\mathscr{S}$. Any ordering of workers in P such that

1. If $p_i \in S \cap T$ and $p_j \notin S \cap T$ then $i < j$.
2. If $p_i \in S \cup T$ and $p_j \notin S \cup T$ then $i < j$.

is an *almost order preserving sequence*. If $|\mathscr{X}| = 2$, then there is always at least one almost order preserving sequence.

The remainder of Section 2.3 is devoted to formally showing that order preserving graphs and graphs with $|\mathscr{X}| = 2$ are the only classes of graphs for which the optimality of HIF can be guaranteed.

2.3.1 HIF is Optimal when $|\mathscr{X}| = 2$

The optimality of HIF has already been proven for order preserving graphs. It only remains to show that HIF is optimal when $|\mathscr{X}| = 2$. Lemma 2.1 is needed to establish that the HIF policy is optimal on instances of OTBMP with $|\mathscr{X}| = 2$.

Lemma 2.1. Let $\mathscr{X} = \{x^{(1)}, x^{(2)}, \ldots, x^{(m)}\}$, and $\mathscr{P} = \{p_1, p_2, \ldots, p_n, p_i'\}$. Consider two OTBMP instances, the first with worker set $P^1 = \mathscr{P} \setminus \{p_i\}$ and the second with worker set $P^2 = \mathscr{P} \setminus \{p_i'\}$. Assume that jobs arrive in the order $x_1, x_2, \ldots, x_n$ in both instances. Suppose that optimal policies for the two instances are labeled $\pi_1^\star$ and $\pi_2^\star$ respectively. Then

$$\left|r(\pi_1^\star, P^1) - r(\pi_2^\star, P^2)\right| \leq 1, \tag{2.13}$$

where r is the function defined by (2.1).

Proof. Let $\tilde{\pi}_2$ be a policy on P^1 defined so that it matches $\pi_2^\star$ on every job assignment, except for the job assigned to p_i by $\pi_2^\star$. Let $\tilde{\pi}_2$ assign this job to p_i' instead. Then

$$r(\pi_2^*, P^2) - 1 \leq r(\tilde{\pi}_2, P^1) \leq r(\pi_1^*, P^1), \tag{2.14}$$

since π_2^* and $\tilde{\pi}_2$ only differ from each other on one job assignment and π_1^* is optimal on P^1. This implies that

$$r(\pi_2^*, P^2) - r(\pi_1^*, P^1) \leq 1. \tag{2.15}$$

Defining $\tilde{\pi}_1$ analogously to $\tilde{\pi}_2$ and exchanging the subscripts in the previous argument gives

$$r(\pi_1^*, P^1) - r(\pi_2^*, P^2) \leq 1. \tag{2.16}$$

Then (2.13) follows from (2.15) and (2.16). QED.

Theorem 2.5 shows that $|\mathscr{X}| = 2$ is sufficient to ensure the optimality of the HIF policy.

Theorem 2.5. Consider an $(\mathscr{X}, P)$-partite graph where $|\mathscr{X}| = 2$ and $(p_1, p_2, \ldots, p_n)$ is an almost order preserving sequence on P. Then the HIF Policy with respect to the ordering $(p_1, p_2, \ldots, p_n)$ yields an optimal assignment for this OTBMP instance.

Proof. We can assume $\mathscr{S} = \{S, T\}$ for some $S, T \subseteq P$.

The proof proceeds by induction on n. For the base case, when there is only one worker and job, any policy will be optimal. Assume the HIF policy is optimal for any instance with fewer than n jobs.

We will partition P into four mutually exclusive regions. Let $R_I \equiv S \cap T$, $R_S \equiv S \setminus T$, $R_T \equiv T \setminus S$, and $R_O \equiv (S \cup T)^{\mathsf{c}}$. Let the HIF policy, π_{HIF}, assign jobs $x_1, x_2, \ldots, x_n$ to regions $R_1, R_2, \ldots, R_n$ and attain a reward value of r_{HIF}. Assume there is another policy, $\pi^\star$, that assigns jobs $x_1, x_2, \ldots, x_n$ to regions $R_1^\star, R_2^\star, \ldots, R_n^\star$ and attains a reward value of $r^\star > r_{\text{HIF}}$. Since $r_{\text{HIF}} \neq r^\star$, there must be some job for which the assigned regions differ between the two policies. Let h be the first such job. Assume without loss of generality that $S_h = S$ (otherwise variables can be renamed by switching S with T). There are five cases to consider:

(1) $R_h = R_T$. The HIF policy will only assign job x_h to region R_O if there are no available workers in S or R_O. Therefore R_T must be the only region with available workers, and $R_h^\star = R_h$, which is a contradiction.

(2) $R_h = R_O$. The HIF policy will only assign job x_h to region R_O if there are no available workers inside S. Therefore regions R_S and R_I are full, and $R_h^\star = R_T$. Since neither policy assigns job x_h to a worker inside S, the two policies must have the same reward at the end of round h. The remaining sets of workers under each policy differ by exactly one, and the differing worker under π_{HIF} is better than the differing worker under $\pi^\star$ (that is, the differing worker under π_{HIF} can handle a larger set of the incoming jobs). Since π_{HIF} behaves optimally on its remaining instance, $r^\star \leq r_{\text{HIF}}$, which is a contradiction.

(3) $R_h = R_S$ and $R_h^\star = R_O$ or R_T. In this case, π_{HIF} assigns job x_h to a worker inside of S, while $\pi^\star$ assigns job x_h to a worker outside of S. Therefore at the end of round h, the reward under π_{HIF} is one more than the reward under $\pi^\star$. By Lemma 2.1, the reward under $\pi^\star$ on its remaining instance is at most one more than the reward under π_{HIF} on its remaining instance. Therefore $r^\star \leq r_{\text{HIF}}$, which is a contradiction.

(4) $R_h = R_S$ and $R_h^\star = R_I$. In this case, both π_{HIF} and $\pi^\star$ have assigned job x_h to a worker inside of S. Therefore the two policies must have the same reward value at the end of round h. The remaining sets of workers under each policy differ by exactly one, and the differing worker under π_{HIF} is better than the differing worker under $\pi^\star$. Since π_{HIF} behaves optimally on its remaining instance, $r^\star \leq r_{\text{HIF}}$, which is a contradiction.

(5) $R_h = R_I$. The HIF policy will only assign job x_h to region R_I if there are no available workers in R_S. Therefore $R^\star \in \{R_O, R_T\}$. Hence π_{HIF} assigns job x_h to a worker inside of S, while $\pi^\star$ assigns job x_h to a worker outside of S. Therefore at the end of round h, the reward under π_{HIF} is one more than the reward under $\pi^\star$. By Lemma 2.1, the

reward under $\pi^\star$ on its remaining instance is at most one more than the reward under π_{HIF} on its remaining instance. Therefore $r^\star \leq r_{\text{HIF}}$, which is a contradiction.

QED.

2.3.2 Necessary Conditions for the Optimality of HIF

In section 2.3.1, we showed that the HIF policy was optimal when $|\mathscr{X}| = 2$. Once again, we are faced with the question of whether order preserving or $|\mathscr{X}| = 2$ is necessary for the optimality of the HIF policy. That is, given any $(\mathscr{X}, P)$-partite graph that is not order preserving and has $|\mathscr{X}| > 2$ an adversary can find some sequence $X = (x_1, x_2, \ldots, x_n)$ for which $x_i \in \mathscr{X}$ for all i and the HIF policy will not be optimal.

We first show that HIF is not optimal if the adversary is allowed to pick the worker set as a subset of P in addition to the incoming job set.

Theorem 2.6. Let G be an $(\mathscr{X}, P)$-partite graph with $\mathscr{X} = \{x^{(1)}, x^{(2)}, \ldots, x^{(m)}\}$ and $m > 2$ that is not order preserving. For any ordering σ_P of P, there exists $P' \subseteq P$, $X' \subset \mathscr{X}$, and some ordering σ_X of X' such that when jobs arrive in order σ_X, the HIF policy (when applied to P' with workers ordered according to σ_P) is not optimal.

Proof. Consider an arbitrary ordering $(p_1, p_2, \ldots, p_n)$ of the elements in P.

Since G is not order preserving and $|\mathscr{X}| > 2$, there is some pair of jobs $x^{(1)}$ and $x^{(2)}$ for which $S_1 \not\subseteq S_2$ and $S_2 \not\subseteq S_1$ (otherwise $S_{i_1} \subseteq S_{i_2} \subseteq \ldots, \subseteq S_{i_n}$ for some $i_1, i_2, \ldots, i_n$, and any ordering which contains the jobs in S_{i_1} first, the jobs in S_{i_2} second, and so on up to the jobs in S_{i_n} is an order preserving sequence). This means that there is some worker $p_i \in S_1 \setminus S_2$ and some worker $p_j \in S_2 \setminus S_1$. In addition, since $|\mathscr{S}| > 2$, there must be some job $x^{(3)}$ for which S_3 is distinct from both S_1 and S_2. There are 3 cases:

1. If S_3 contains both p_i and p_j, then let $P' = \{p_i, p_j\}$. If $i < j$, then consider the instance in which jobs arrive in order $(x^{(3)}, x^{(2)})$. Since $x^{(3)}$ is adjacent to both p_i and p_j and $j > i$, then the HIF policy will assign job $x^{(3)}$ to worker p_j. This leaves job $x^{(2)}$ to be assigned to worker p_i. The reward (given by (2.1)) under this assignment is 1, whereas swapping the two assignments gives a reward of 2. This means that the HIF policy is not optimal. If $i > j$, then a similar argument shows that the HIF policy is not optimal when jobs arrive in order $(x^{(3)}, x^{(1)})$.

2. If S_3 contains neither p_i nor p_j, then then let $P' = \{p_i, p_j\}$. If $i < j$, then consider the instance in which jobs arrive in order $(x^{(3)}, x^{(2)})$. Since $x^{(3)}$ is not adjacent to either

p_i or p_j and $j > i$, then the HIF policy will assign job $x^{(3)}$ to worker p_j. This leaves job $x^{(2)}$ to be assigned to worker p_i. The reward under this assignment is 0, whereas swapping the two assignments gives a reward of 1. This means that the HIF policy is not optimal. If $i > j$, then a similar argument shows that the HIF policy is not optimal when jobs arrive in order $(x^{(3)}, x^{(1)})$.

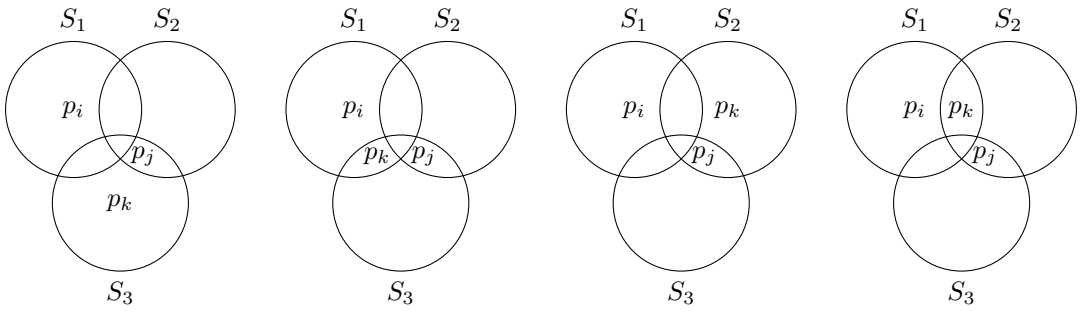
3. Suppose S_3 contains exactly one of p_i and p_j. Assume without loss of generality that $p_j \in S_3$ and $p_i \notin S_3$. Then there must be some worker, p_k, that distinguishes S_3 from S_2. Figure 2.3 shows the possibilities for p_k.

   (a) If $p_k \in S_3 \setminus (S_2 \cup S_1)$, then let $P' = \{p_i, p_k\}$. If $i < k$, let jobs arrive in order $(x^{(2)}, x^{(3)})$. Since $x^{(2)}$ is not adjacent to either p_i or p_k and $k > i$, then the HIF policy will assign job $x^{(2)}$ to worker p_k. This leaves $x^{(3)}$ to be assigned to worker p_i. The reward under this assignment is 0, while swapping the two assignments gives a reward of 1. This means that the HIF policy is not optimal. If $i > k$, then a similar argument shows that the HIF policy is not optimal when jobs arrive in order $(x^{(2)}, x^{(1)})$.

   (b) If $p_k \in S_3 \cap S_1$ but $p_k \notin S_2$, then let $P' = \{p_j, p_k\}$. If $j < k$, let jobs arrive in order $(x^{(3)}, x^{(1)})$. Since $x^{(3)}$ is adjacent to both p_j and p_k and $k > j$, then the HIF policy will assign job $x^{(3)}$ to worker p_j. This leaves job $x^{(1)}$ to be assigned to worker p_j. The reward under this assignment is 1, while swapping the two assignments gives a reward of 2. This means that the HIF policy is not optimal. If $k < j$, then a similar argument shows that the HIF policy is not optimal when jobs arrive in order $(x^{(2)}, x^{(3)})$.

   (c) If $p_k \in S_2 \setminus (S_1 \cup S_3)$, then let $P' = \{p_i, p_k\}$. If $i < k$, let jobs arrive in order $(x^{(3)}, x^{(2)})$. Since $x^{(3)}$ is not adjacent to either p_i or p_k and $k > i$, then the HIF policy will assign job $x^{(3)}$ to worker p_k. This leaves job $x^{(2)}$ to be assigned to worker p_i. The reward under this assignment is 0, while swapping the two assignments gives a reward of 1. This means that the HIF policy is not optimal. If $i > k$, then a similar argument shows that the HIF policy is not optimal when jobs arrive in order $(x^{(3)}, x^{(1)})$.

   (d) If $p_k \in S_1 \cup S_2$ but $p_k \notin S_3$, then let $P' = \{p_j, p_k\}$. If $j < k$, let jobs arrive in order $(x^{(2)}, x^{(1)})$. Since $x^{(2)}$ is adjacent to both p_j and p_k and $k > j$, then the HIF policy will assign job $x^{(2)}$ to worker p_k. This leaves job $x^{(1)}$ to be assigned to worker p_j. The reward under this assignment is 1, whereas swapping the two assignments gives a reward of 2. This means that the HIF policy is not optimal.



(a) $p_k \in S_3 \setminus (S_2 \cup S_1)$ (b) $p_k \in (S_3 \cap S_1) \setminus S_2$ (c) $p_k \in S_2 \setminus (S_1 \cup S_3)$ (d) $p_k \in (S_1 \cap S_2) \setminus S_3$

Figure 2.3: Possibilities for p_k.

If $j > k$, then a similar argument shows that the HIF policy is not optimal when jobs arrive in order $(x^{(2)}, x^{(3)})$.

QED.

Note that in the proof of Theorem 2.6 $|X'| = |P'| = 2$. This will allow us to show that HIF is necessary in the sense described at the beginning of this section.

Corollary 2.1. Let $G = (V, E)$ be an $(\mathscr{X}, P)$-partite graph that is not order preserving and has $|\mathscr{X}| > 2$. For any ordering σ_P of P, there exists $X \subset \mathscr{X}$, and some ordering σ_X of X such that when jobs arrive in order σ_X, the HIF policy is not optimal.

Proof. Assume the instance constructed by the Proof of Theorem 2.6 had $P' = \{p_j, p_k\}$ and $X' = \{x_k, x_\ell\}$ with jobs arriving in order x_k, x_ℓ. Then the adversary can construct X by sending in copies of x_k until exactly one job in P' is occupied, and then send in copies of x_ℓ until all workers have been filled.

Then the assignment to workers p_j and p_ℓ will be the same as when HIF was performed on X' and P'. Since, using the arguments provided in the proof Theorem 2.6, the policy that made the same assignments except for switching the jobs assigned to p_j and p_ℓ is one better than the assignment given by HIF, HIF is not optimal. QED.

2.4 CONCLUSIONS AND FUTURE DIRECTIONS

Matching problems have been well studied in the literature, given the breadth of real-world problems that they model. This chapter introduces and studies the Online Total Bipartite Matching Problem, providing tight bounds on the competitive ratio for more general versions of the problem. It also formalizes the intuitive policy on order preserving graphs as the HIF policy, and rigorously shows exactly when this policy is guaranteed to be optimal.

While OTBMP may be too simple for most real world applications, we believe that this work can provide the foundation for a more realistic model then those previously studied. Extending these results to more complex models is one direction for future research. For example, we could adjust our model to accommodate ongoing arrivals, with workers available to be reused after completing a job. We could also consider OTBMP from a more theoretical perspective, and extend these results to more general classes graphs or different arrival models (such as random arrivals or stochastic arrivals).

CHAPTER 3: REUSABLE SEQUENTIAL STOCHASTIC ASSIGNMENT PROBLEM

Recall that a second problem with using standard SSAP to model real world problems, and in particular, the aviation security passenger screening problem, is that in traditional SSAP, once a worker is used, it cannot be used again. To solve this complaint, this chapter introduces a variation of SSAP, titled the Reusable Sequential Stochastic Assignment Problem (RSSAP). In traditional SSAP, once a worker receives a job assignment, it cannot be used for the remainder of the instance. In RSSAP, each job is associated with a length of time. When a worker receives a job assignment, it will finish processing the job afcer the given length of time, and will then be available to receive new job assignments. The chapter is organized as follows. Section 3.1 describes the problem formulation. Section 3.2 proposes approximation algorithms for RSSAP where the job values are IID. Section 3.3 assumes no prior information on job values and proposes an algorithm for RSSAP when jobs have a random order of arrivals. Section 3.4 presents a summary of the results and provides concluding remarks.

Note that this work was done jointly with Ge Yu, and much of it also appears in Chapter 5 of her thesis [32].

3.1 RSSAP FORMULATION

This section describes the stochastic online matching problem with reusable resources in the setting of sequential stochastic assignment problems with reusable workers (RSSAP). The model is described using the classic SSAP setting with workers and jobs, which may be extended to other scenarios including interval scheduling problems and online bidding problems.

Let M denote the total number of workers and $\{w_m\}$ denote the success rate for each worker, with $w_1 \leq w_2 \leq \ldots \leq w_M$. A worker who is assigned to a job and has not completed that job is said to be *busy*; otherwise, the worker is said to be *available*. Each arriving job has a three-dimensional vector revealed upon arrival, with the first component representing the arrival time, the second component representing the job value and the third component representing the job length. For example, if a job length is 2 and a worker is assigned to the job at time $t = 1$, the worker becomes available again at time $t = 3$. Let $\mathbf{j}_n = (a_n, v_n, l_n)$ denote the three-dimensional vector for the n^{th} job arrival, $n = 1, 2, \ldots$. Let $a(J)$, $v(J)$ and $l(J)$ represent the arrival time, the value and the length of job J, respectively. We study the special case where all jobs have the same length, denoted by l_0. Let $F_v(v)$ denote the

cumulative distribution function (cdf) for job values.

In RSSAP, the time axis is divided into *slots*, indexed by $t = 1, 2, \ldots, T$. Jobs are assumed to arrive at the beginning of the time slot, and are assigned to one of the available workers or discarded immediately. Moreover, we assume that jobs will be completed at the end of time slots. Note that the assumption of jobs being completed at the end of time slots is superfluous, since workers who complete a job in the middle of a slot cannot be assigned to any job until the beginning of the next time slot. A job arrives at the beginning of each time slot with probability p. This job arrival process is defined as *geometric arrivals*, since the inter-arrival times are IID geometric random variables. The geometric arrival process is the discrete-time counterpart for a Poisson process. If a time slot has no job arrival, we assume that there is a virtual job with value zero and length same as real jobs arriving at the beginning of this slot [3]. Therefore, each job keeps the assigned worker busy over the time period $[a_n, a_n + l_n)$ ($a_n \in \mathbb{Z}^+$ and $l_n \in \mathbb{Z}^+ \bigcup \{0\}$ for all n).

We consider a sequence of jobs that arrives during the time interval $[1, T]$. Define a sequence of assignment variables, $X_{n,m} \in \{0, 1\}$, as an indicator of the n^{th} job assigned to worker w_m (we use the success rate to refer to a worker). Job assignments are assumed to be irrevocable and non-preemptive. Moreover, a job can be assigned to at most one worker, and a worker can be assigned to at most one job at a time. An algorithm $\mathcal{A}$ defines a sequence of assignment variables, $\{X^{\mathcal{A}}_{n,m}\}$ for $n = 1, 2, \ldots, T$ and $m = 1, 2, \ldots, M$.

The objective of RSSAP is to maximize the expected *reward* for assigning all the jobs using available workers, where the expectation is with respect to the distribution of job sequences and the randomization of the algorithm (for randomized algorithm only). The reward for assigning job J_n using algorithm $\mathcal{A}$, denoted by $R_{\mathcal{A}}(J_n)$, is given by the product of the job values and the assigned workers' success rates,

$$R_{\mathcal{A}}(J_n) = \sum_{m=1}^{M} X^{\mathcal{A}}_{n,m} v_n w_m. \tag{3.1}$$

Dynamic programming can be applied to obtain the optimal offline algorithm for a given job sequence. However, dynamic programming is intractable due to the state space dimension [33]. Therefore, we seek approximation algorithms and compare our algorithms to the value of the optimal online algorithm.

Definition 3.1. Let OPT denote the optimal (maximal) expected reward for assigning job sequences in RSSAP. Let $R(\mathcal{A})$ denote the reward for assigning such job sequences using

algorithm $\mathcal{A}$. Then algorithm $\mathcal{A}$ is said to be a $\gamma_{\mathcal{A}}$-approximation if

$$\gamma_{\mathcal{A}} \geq \frac{\mathbb{E}[OPT]}{\mathbb{E}[R(\mathcal{A})]}, \tag{3.2}$$

where the expectation in the numerator is taken with respect to the distribution of job sequences, while the expectation in the denominator is taken with respect to the distribution of job sequences and the randomization of $\mathcal{A}$ (for randomized algorithm only).

3.2 APPROXIMATION ALGORITHMS FOR RSSAP WITH IID JOB ARRIVALS

This section provides approximation algorithms for RSSAP with IID job arrivals. Two cases are considered: (a) all jobs have the same fixed length and IID values with distribution $F_v(v)$; (b) all jobs have IID memoryless length and IID values with distribution $F_v(v)$. The proposed approximation algorithms for both cases are Greedy SSAP optimal policies.

3.2.1 Preliminary: Classic SSAP Policy

[19] introduce the sequential stochastic assignment problem (SSAP), where T workers with success rates $\tau_1 \leq \tau_2 \leq \ldots \leq \tau_T$ are assigned to T sequentially arriving jobs with values $\{\mathcal{C}_t\}_{t=1}^T$ (random variables) revealed upon arrival. This problem is referred to as the *T-depth* SSAP problem. The objective is to maximize the total expected reward $\mathbb{E}[\sum_{t=1}^T \tau_{j_t}\mathcal{C}_t]$, where j_t is the index of the worker assigned to the t^{th} job with value $\mathcal{C}_t$. The optimal policy uses recursive equations to compute threshold values for each job assignment, which motivates our proposed approximation algorithms for RSSAP.

Theorem 3.1 ([19])**.** For the t^{th} job arrival with job value $\mathcal{C}_t$, there are $T-t+1$ workers available, $t = 1, 2, \ldots, T$. The thresholds for $\mathcal{C}_t$ are given by $-\infty = a_0^{T-t} \leq a_1^{T-t} \leq \ldots \leq a_{T-t+1}^{T-t} = +\infty$, with

$$a_i^{T-t} = \int_{a_{i-1}^{T-t-1}}^{a_i^{T-t-1}} x dF_{\mathcal{C}}(x) + a_{i-1}^{T-t-1} F_{\mathcal{C}}(a_{i-1}^{T-t-1}) \tag{3.3}$$

$$+ a_i^{T-t-1}(1 - F_{\mathcal{C}}(a_i^{T-t-1})),\ i = 1, 2, \ldots, T-t, \tag{3.4}$$

where $F_{\mathcal{C}}(x)$ is the distribution for job values. If $\mathcal{C}_t \in (a_{i-1}^{T-t}, a_i^{T-t}]$, then the worker with the i^{th} smallest success rate among the $T-t+1$ available workers is assigned to the t^{th} job under the optimal policy (referred to as the SSAP optimal policy). Moreover, a_i^{T-t} is the

expected job value that is assigned to the worker with i^{th} smallest success rate among the $T-t$ available workers for $i = 1, 2, \ldots, T-t$.

[3] consider a Generalized SSAP problem (GSSAP), where at each time t a job arrives with some probability $0 < p \leq 1$. Therefore, the total number of arriving jobs is a random variable. They show that this GSSAP is equivalent to an SSAP with cdf given by

$$F_G(x) = (1-p) + pF_{\mathcal{C}}(x). \tag{3.5}$$

That is, no job arrival is treated as an arriving job with value zero. The optimal policy for the GSSAP is given by Theorem 3.1 with $F_G(x)$ substituted for $F_{\mathcal{C}}(x)$. $F_G(x)$ is referred to the *refined value distribution*.

We consider equal-length job sequences. Let $l_0 \in \mathbb{Z}^+ \bigcup \{0\}$ denote the length of each job. If $l_0 \leq 1$, then the optimal solution is to assign every arriving job to the worker with the largest success rate, who is able to complete all jobs. We assume $l_0 \geq 2$ in the following analysis. Sections 3.2.2 and 3.2.3 consider a single available worker while Section 3.2.4 discusses the case of multiple workers.

Several definitions are needed. The time interval starting from the arrival time (included) of an assigned job until its completion time (not included) is referred to as the *blocking window* of the assigned job. Jobs arriving after the assigned job and during this blocking window are *blocked* by this assigned job (the assigned job is in its own blocking window) since assignments are irrevocable and non-preemptive. Therefore, only one job can be assigned within each blocking window. In the case of IID arrivals, all blocking windows are deterministic with the same size of l_0.

3.2.2 A Single Worker

Since only one worker is available, we assume that $w_1 = 1$. We divide the time axis into *stages*, which are time intervals of length $(2l_0 - 1)$ and are numbered sequentially. For example, the time interval starting from $t = 1$ and lasting until $t = 2l_0$ (i.e., $[1, 2l_0)$) is referred to as *stage one*, the time interval from $t = 2l_0$ to $t = 4l_0 - 1$ (i.e., $[2l_0, 4l_0 - 1)$) is referred to as *stage two*, and so forth. We propose the Greedy Threshold algorithm, which is a threshold algorithm based on the optimal policy for SSAP.

The intuition behind the Greedy Threshold algorithm is as follows: (1) the worker should be used as many times as possible, and (2) the worker should be assigned to a job with the highest value (in expectation) whenever the worker is available. According to the Greedy

Algorithm 3.1 Greedy Threshold Algorithm

1: Compute the refined cdf $F_G(v)$ for job values using (3.5).
2: Compute the threshold values in a l_0-depth SSAP problem with job value distribution $F_G(v)$, $\{a_i^j\}$. Then a_i^j is the expected value of the i^{th} smallest job value among j IID jobs with cdf $F_G(v)$, for $i = 1, 2, \ldots, j$ and $j = 1, 2, \ldots, l_0$.
3: Beginning at stage one (i.e., from $t = 1$ to $t = 2l_0$).
4: **while** $t \leq T$ **do**
5: Re-index the time slots in each stage as $t' = 1$ to $t' = 2l_0$.
6: If a job J arrives at time t', then J will be assigned to worker w_1 if and only if

$$v(J) \geq a_{l_0-t'}^{l_0-t'}, \tag{3.6}$$

for $t' = 1, 2, \ldots, l_0$ with $a_0^0 = 0$.
7: If worker w_1 is assigned a job, let t^* denote the arrival time of the job. Then the job will be completed at $t = t^* + l_0$. Therefore, the next stage is defined as starting from $t = t^* + l_0$ until $t = t^* + 3l_0 - 1$. Otherwise, let t^* denote the re-indexed time $t' = l_0$, and the next stage is defined as starting from $t = t^* + 1$ until $t = t^* + 2l_0$.
8: **end while**

Threshold algorithm, the worker is used at least once every $(2l_0 - 1)$ time slots and assigned to the job with the highest value among blocked jobs, in expectation.

To compute the approximation ratio of the Greedy Threshold algorithm, we first derive an upper bound for the optimal expected reward. Then we give a lower bound for the expected reward using the Greedy Threshold algorithm. Note that the Greedy Threshold algorithm is a deterministic algorithm, and hence, the expectation is taken over the distribution of the job sequence.

Lemma 3.1. An upper bound for the optimal expected reward for assigning equal-length jobs to a single reusable worker, R_E^*, is

$$R_E^* \leq (\lfloor \frac{T-1}{l_0} \rfloor + 1) a_{l_0}^{l_0}, \tag{3.7}$$

where T is the arrival time of the last job, l_0 is the job length, and $a_{l_0}^{l_0}$, defined by (3.3), is the expectation of the largest job value for l_0 IID job values.

Proof. The proof uses properties of the IID arrivals of the job sequence. We provide upper bounds for two elements: (1) the total number of completed job assignments; (2) the expected job value for each job assignment. The product of upper bounds for these two elements provides an upper bound for the optimal expected reward.

Consider the total number of job assignments the worker is able to complete. Since each job will take l_0 time slots to complete, the total number of completed assignments is at most

$(\lfloor (T-1)/l_0 \rfloor + 1)$, which is achieved by assigning jobs to the worker back-to-back (the last assignment may be completed after T).

Consider the expected job value assigned to the worker. Each time a job is assigned to the worker, the job will incur a blocking window of length l_0. Since job arrivals in the entire job sequence are IID, each blocking window of length l_0 has the same distribution, and hence, the optimal reward earned by any assignment can be upper bounded by the expected reward earned by an optimal online algorithm trying to pick the largest job from a stream of l_0 jobs. Using proof techniques from [19]'s proof of Theorem 3.1, we can show that this upper bound is equal to $a_{l_0}^{l_0}$. In particular, we use induction on l_0. For the base case, when $l_0 = 1$, the optimal algorithm is to pick the singular job, resulting in a reward of

$$\mathbb{E}_{X \sim F_G}[X] = \int_{-\infty}^{\infty} x dF_G(x) = a_1^1. \tag{3.8}$$

Assume the algorithm with he highest expectation on $l_0 - 1$ jobs has expectation $a_{l_0-1}^{l_0-1}$. Then consider attempting to maximize the expectation over l_0 jobs. The clear optimal strategy is to pick the first job if it's expectation is higher than $a_{l_0-1}^{a} l_0 - 1$, or to discard it otherwise. Then in expectation, this produces a reward of

$$\mathbb{P}_{X \sim F_G}[X > a_{l_0-1}^{l_0-1}] \mathbb{E}_{X \sim F_G}[X | X > a_{l_0-1}^{l_0-1}] + \mathbb{P}_{X \sim F_G}[X \leq a_{l_0}^{l_0}] a_{l_0}^{l_0} \tag{3.9}$$

which can be written as

$$\int_{a_{l_0-1}^{l_0-1}}^{\infty} x dF_g(x) + a_{l_0-1}^{l_0-1} F_g(a_{l_0-1}^{l_0-1}) = a_{l_0}^{l_0}. \tag{3.10}$$

Therefore by induction, the optimal algorithm that tries to maximize the expected job value chosen from a stream of l_0 jobs earns an expected reward of $a_{l_0}^{l_0}$.

The product of these two upper bounds gives an upper bound for the optimal expected reward. QED.

Proposition 3.1 provides a lower bound for the expected reward using the Greedy Threshold algorithm.

Proposition 3.1. A lower bound for the expected reward using the Greedy Threshold algorithm, $\mathbb{E}[R_{GT}]$, is

$$\mathbb{E}[R_{GT}] \geq \lfloor \frac{T-1}{2l_0 - 1} \rfloor a_{l_0}^{l_0}, \tag{3.11}$$

where the expectation is taken over the distribution of the job sequence.

Proof. Since the Greedy Threshold algorithm treats each stage the same way and job arrivals are IID, the expected reward using the Greedy Threshold algorithm is the product of the number of stages and the expected reward in each stage.

First, we consider the number of stages. From the Greedy Threshold algorithm, the worker is assigned once in each stage. Therefore, the total number of stages is the total number of jobs completed by the worker, denoted by $N(S)$. Note that the actual size of a stage is less than or equal to $(2l_0 - 1)$, and hence, $N(S)$ has a lower bound given by

$$N(S) \geq \lfloor \frac{T-1}{2l_0 - 1} \rfloor. \tag{3.12}$$

Next, we consider the expected reward for the Greedy Threshold algorithm in each stage. Since the threshold values for the l_0-depth SSAP problem are used for assigning jobs to the worker, then from Theorem 3.1, the expected reward for each stage is $a_{l_0}^{l_0}$.

Combining the number of stages and the expected reward in each stage together leads to the desired result. QED.

Theorem 3.2 provides the approximation ratio of the Greedy Threshold algorithm on equal-length job sequences.

Theorem 3.2. The Greedy Threshold algorithm is asymptotically a $(2 - \frac{1}{l_0})$-approximation for IID equal-length job sequences, where l_0 is the job length.

Proof. The result follows directly from Lemma 3.1 and Proposition 3.1. In particular, let γ_{GT} denote the approximation ratio of the Greedy Threshold algorithm. Then,

$$\gamma_{GT} = \frac{R_E^*}{\mathbb{E}[R_{GT}]} \leq \frac{(\lfloor \frac{T-1}{l_0} \rfloor + 1) a_{l_0}^{l_0}}{\lfloor \frac{T-1}{2l_0-1} \rfloor a_{l_0}^{l_0}} \leq \frac{\frac{T-1}{l_0} + 1}{\frac{T-1}{2l_0-1} - 1} \to 2 - \frac{1}{l_0}, \tag{3.13}$$

as $T \to +\infty$. QED.

Theorem 3.2 holds for any job value distribution $F_v(v)(F_G(v))$ with finite mean. Moreover, Theorem 3.2 holds for jobs with non-integer (continuous) lengths, as long as job arrivals only happen at the beginning of time slots.

3.2.3 Fixed-threshold Algorithm for a Single Worker

This section considers a class of Greedy algorithms using a single fixed threshold, as a comparison with the Greedy Threshold algorithm proposed in Section 3.2.2.For general distributions of job values, the approximation ratio of the Greedy algorithm with a fixed

threshold (referred to as the Fixed-threshold algorithm) does not have a simple closed-form expression. However, if the job value follows a uniform distribution, the expression for the approximation ratio of the Fixed-threshold algorithm can be obtained.

Let $\hat{v}$ denote the threshold for the job value under the Fixed-threshold algorithm: if $v(J) \geq \hat{v}$ and the worker is available, assign job J to the worker; otherwise, discard the job. We will determine the value of $\hat{v}$ later. To analyze the Fixed-threshold algorithm, divide the time axis into stages: the time interval starting from the time slot when the worker is first available after completing a previous job until the time slot when the worker completes one job assignment. For example, the first stage starts from time $t = 1$ until the time when the worker completes the first job assignment, denoted by t_1. Then the second stage starts from t_1 until the time when the worker completes the second job assignment. That is, one job is completed in each stage.

Let L_i denote the length of stage i, for $i = 1, 2, \ldots, N_S$, where N_S denotes the total number of stages that have occurred by time T (the last stage may be completed after T). Let R_i denote the reward achieved by the Fixed-threshold algorithm in stage i. Then, $\{L_i\}$ and $\{R_i\}$ are both IID random variables, since the Fixed-threshold algorithm uses the same threshold value for all job assignments. Moreover, each stage consists of two parts: (1) the worker waits to be assigned in the first part, whose length is a geometrically distributed random variable with parameter p_L given by

$$\begin{aligned} p_L &= \mathbb{P}(\text{the worker is assigned at a time slot } t) && (3.14)\\ &= \mathbb{P}\left(\text{job } J \text{ is assigned to the worker} \mid \text{job } J \text{ arrives}\right)\mathbb{P}\left(\text{job } J \text{ arrives}\right) && (3.15)\\ &= p\mathbb{P}(v(J) \geq \hat{v}); && (3.16) \end{aligned}$$

and (2) the worker completes the assigned job in the second part, whose length is a constant l_0. Therefore,

$$\mathbb{E}[L_i] = \frac{1}{p\mathbb{P}(v(J) \geq \hat{v})} + l_0 - 1, \tag{3.17}$$

where the minus one is because the geometric distribution of the first part starts from zero. The expectation of R_i is given by $\mathbb{E}[R_i] = \mathbb{E}[v(J) \mid v(J) \geq \hat{v}]$, where the expectation is taken with respect to the original value distribution $F_v(v)$.

Since the number of stages that have occurred by any time t is a renewal process, N_S is a stopping rule for both $\{L_i\}$ and $\{R_i\}$. Let R_{FT} denote the reward using the Fixed-threshold

algorithm for equal-length jobs. By Wald's identity,

$$\mathbb{E}[\sum_{i=1}^{N_S} L_i] = \mathbb{E}[N_S]\mathbb{E}[L_i] \geq T \Rightarrow \mathbb{E}[N_S] \geq \frac{T}{\mathbb{E}[L_i]}, \tag{3.18}$$

$$\mathbb{E}[R_{FT}] = \mathbb{E}[\sum_{i=1}^{N_S} R_i] = \mathbb{E}[N_S]\mathbb{E}[R_i] \geq T\frac{\mathbb{E}[R_i]}{\mathbb{E}[L_i]}. \tag{3.19}$$

Therefore, maximizing the lower bound for $\mathbb{E}[R_{FT}]$ is the same as maximizing the ratio $\mathbb{E}[R_i]/\mathbb{E}[L_i]$. Substituting (3.17) into (3.19) leads to the following optimization problem:

$$\max_{\hat{v}\in[v_{min},v_{max}]} \frac{\mathbb{E}[v(J) \mid v(J) \geq \hat{v}]}{\frac{1}{p\mathbb{P}(v(J)\geq\hat{v})} + l_0 - 1}, \tag{3.20}$$

where $[v_{min}, v_{max}]$ is the support for the job value. To obtain the optimal threshold $\hat{v}$, take the derivative of (3.20) with respect to $\hat{v}$ and set the derivative to zero,

$$\frac{\hat{v}}{(\mathbb{E}[v(J)|v(J)\geq\hat{v}] - \hat{v})\mathbb{P}(v(J)\geq\hat{v})} = p(l_0 - 1). \tag{3.21}$$

In general, there is no closed-form solution to (3.21), and hence, the specific choice of a threshold for job values $\hat{v}$ depends on parameters of the distribution. We take uniform distributions and exponential distributions as examples.

If the job value is uniformly distributed on $[A, B]$ (i.e., $F_v(v) = (v - A)/(B - A)$ for $v \in [A, B]$), then for $A \leq \hat{v} \leq B$, (3.21) becomes

$$\frac{\hat{v}}{\left(\frac{B+\hat{v}}{2} - \hat{v}\right)\left(\frac{B-\hat{v}}{B-A}\right)} = p(l_0 - 1). \tag{3.22}$$

This can be rewritten as

$$p(l_0 - 1)\hat{v}^2 - 2(p(l_0 - 1)B + B - A)\hat{v} + p(l_0 - 1)B^2 = 0 \tag{3.23}$$

Therefore

$$\hat{v} = \frac{2(p(l_0 - 1)B + B - A) \pm \sqrt{4(p(l_0 - 1)B + B - A)^2 - 4p^2(l_0 - 1)^2B^2}}{2p(l_0 - 1)} \tag{3.24}$$

$$= \frac{(l_0 - 1)pB + (B - A) \pm \sqrt{2(B - A)(l_0 - 1)pB + (B - A)^2}}{(l_0 - 1)p} \tag{3.25}$$

Note that if we add the discriminant, the resulting value is larger than B, but we only

only want to consider $\hat{V} \in [A, B]$. Therefore we use the root that subtracts the discriminant, giving a final threshold of

$$\hat{v} = \frac{(l_0 - 1)pB + (B - A) - \sqrt{2(B - A)(l_0 - 1)pB + (B - A)^2}}{(l_0 - 1)p}. \tag{3.26}$$

To compute a lower bound on the expected reward, we note that since value computed above is optimal, using other value from $[A-B]$ for $\hat{v}$ will produce a smaller expected reward, and therefore is a lower bound on the optimal reward. In particular, if we substitute $\hat{v} = A$ into (3.19) (where $\mathbb{E}[R_i]/\mathbb{E}[L_i]$ is given by (3.20)), we get

$$\mathbb{E}[R_{FT}] \geq T \frac{\mathbb{E}[v(J) \mid v(J) \geq \hat{v}]}{\frac{1}{p\mathbb{P}(v(J) \geq \hat{v})} + l_0 - 1} \tag{3.27}$$

$$= T \frac{(1/2)(B + \hat{v})}{\frac{B-A}{p(B-\hat{v})} + l_0 - 1} \tag{3.28}$$

$$\geq T \frac{(1/2)(B + A)}{\frac{B-A}{p(B-A)} + l_0 - 1} \tag{3.29}$$

$$= T \frac{p}{2} \frac{B + A}{1 + p(l_0 - 1)} \tag{3.30}$$

We would like to compare this to the optimal expected reward for jobs with values uniformly distributed on $[A, B]$, which is upper bounded by

$$R_E^{U\star} \leq (\lfloor (T - 1)/l_0 \rfloor + 1) \left(B - \frac{B - A}{l_0 + 1} \right), \tag{3.31}$$

where the first factor corresponds to the maximum number of jobs that could be assigned and the second factor is the expected value of the largest value when l_0 values are selected from a uniform distribution on $[A, B]$, which is an upper bound on the expected value of any single job in a blocking window of length l_0.

Let γ_{FT}^U denote the approximation ratio of the Fixed-threshold algorithm. Then combining (3.30) and (3.31) leads to

$$\gamma_{FT}^U = \frac{R_E^{U*}}{\mathbb{E}[R_{FT}]} \tag{3.32}$$

$$\leq \frac{l_0 B + A}{l_0(l_0 + 1)} \frac{2(1 + (l_0 - 1)p)}{p(B + A)} \frac{T + l_0 - 1}{T} \tag{3.33}$$

$$\leq 2 \frac{1 + (l_0 - 1)p}{(l_0 + 1)p} \frac{T + l_0 - 1}{T}. \tag{3.34}$$

Note that $1 + (l_0 - 1)p \leq (l_0 + 1)p$ for $p \geq 1/2$. Therefore, the asymptotic approximation ratio of the Fixed-threshold algorithm is less than 2 for $p \geq 1/2$.

3.2.4 Multiple Workers

This section extends the results in Section 3.2.2 to the case of multiple workers. Suppose that there are M workers available, with success rates $w_1 \leq w_2 \leq \ldots \leq w_M$. Similar to the case of a single worker, the time axis is divided into stages of length $(2l_0 - 1)$.

If $M < l_0$, these workers may not be able to complete all jobs in a sequence. On the other hand, if $M \geq l_0$, there will always be redundant workers, and workers with the l_0 largest success rates are able to complete all the jobs in a job sequence. Lemma 3.2 gives an upper bound for the optimal expected reward using multiple workers.

Lemma 3.2. An upper bound for the optimal expected reward for assigning equal-length job sequences to multiple workers, denoted by R^*_M, is

$$R^*_M \leq (\lfloor (T-1)/l_0 \rfloor + 1) \sum_{i=l_0-\min\{M,l_0\}+1}^{l_0} a_i^{l_0} w_{M-l_0+i}, \tag{3.35}$$

where M is the number of workers with success rates $w_1 \leq w_2 \leq \ldots \leq w_M$, T is the arrival time of the last job, l_0 is the common job length, and $\{a_i^{l_0}\}$, defined by (3.3), are the expectations of the order statistics of l_0 IID job values with cdf $F_G(v)$, for $i = 1, 2, \ldots, l_0$.

Proof. The proof is similar to the proof of Lemma 3.1. We consider an easier problem where every l_0 rounds (which we will call a stage), all workers are released and can accept new jobs. Notice that any solution to RSSAP is also a solution to this new problem, and therefore an upper bound for the new problem is also an upper bound for RSSAP. We prove the upper bound by providing upper bounds for two elements: (1) the number of job assignments each worker can complete, and (2) the expected job value that is assigned to each worker in a stage.

Since each worker can only complete one job in each stage, an upper bound for the number of jobs each worker can complete is $(\lfloor (T-1)/l_0 \rfloor + 1)$. Since only M workers are available, the optimal expected reward obtainable during any time interval of length l_0 has an upper bound given by the optimal reward for assigning the l_0 jobs to $\min\{M, l_0\}$ workers with the largest success rates. Since each stage is an example of a standard SSAP problem, from Theorem 3.1, an upper bound for the optimal expected reward during any time interval of

length l_0 is $\sum_{i=l_0-\min\{l_0,M\}+1}^{l_0} a_i^{l_0} w_{M-l_0+i}$. Combining these two upper bounds leads to the desired result. QED.

The algorithm for RSSAP with multiple workers, the *Greedy SSAP-stage algorithm*, is similar to the Greedy Threshold algorithm. The time axis is divided into $(2l_0 - 1)$-length stages, and each worker (if $M > l_0$, then only workers with the l_0 largest success rates are used) is assigned to one job in each stage. At the beginning of each stage, all workers complete previously assigned jobs and become available. The only difference is that workers have their own threshold values, which is a result of the heterogeneity of workers and Hardy's Lemma [34]. The l_0-depth SSAP optimal policy is applied in the first half of each stage: the worker with the largest success rate is assigned to the job with the largest value (in expectation) out of l_0 IID jobs (i.e., jobs in a blocking window); the worker with the second largest success rate is assigned to the job with the second largest value (in expectation) out of l_0 IID jobs; and so on. For simplicity, if $l_0 > M$, add $l_0 - M$ virtual workers with success rates zero and refer the virtual and original workers together as workers with success rates $w'_1 \leq w'_2 \leq \ldots w'_{l_0}$. Otherwise, if $l_0 \leq M$, only use workers with the l_0 largest success rates and refer them as workers with success rates $w'_1 \leq w'_2 \leq \ldots \leq w'_{l_0}$.

Algorithm 3.2 Greedy SSAP-stage Algorithm

1: Compute the refined cdf $F_G(v)$ for job values using (3.5).
2: Compute the threshold values in a l_0-depth SSAP problem with job value distribution $F_G(v)$, $\{a_i^j\}$. Then a_i^j is the expected value of the i^{th} smallest job value among j IID jobs with cdf $F_G(v)$, for $i = 1, 2, \ldots, j$ and $j = 1, 2, \ldots, l_0$.
3: Beginning at stage one (i.e., from $t = 1$ to $t = 2l_0$).
4: **while** $t \leq T$ **do**
5: Reindex the time slots in each stage as $t' = 1$ to $t' = 2l_0$.
6: At time t', there are workers $w_1^{t'} \leq w_2^{t'} \leq \ldots \leq w_{l_0-t'+1}^{t'}$ available. If a job J arrives, then J is assigned to worker $w_j^{t'}$ if and only if

$$a_{j-1}^{l_0-t'} \leq v(J) < a_j^{l_0-t'}, \tag{3.36}$$

for $t' = 1, \ldots, l_0$ with $a_0^k = 0$ for $k = 1, 2, \ldots, l_0 - 1$.
7: At time $t_c = 2l_0$, all workers are available again. Therefore, the next stage is defined as starting from $t = t_c$ till $t = t_c + 2l_0$.
8: **end while**

Proposition 3.2. A lower bound for the expected reward using the Greedy SSAP-stage

algorithm, denoted by $\mathbb{E}[R_{GSS}]$, is

$$\mathbb{E}[R_{GSS}] \geq \lfloor \frac{T-1}{2l_0-1} \rfloor \sum_{i=l_0-\min\{l_0,M\}+1}^{l_0} a_i^{l_0} w_{M-l_0+i}, \tag{3.37}$$

where the expectation is taken over the distribution of the job sequence.

Proof. The proof is similar to the proof of Proposition 3.1. Since job values are IID and the Greedy SSAP-stage algorithm assigns jobs arriving in each stage using the same rule, by Wald's identity, the expected reward using the Greedy SSAP-stage algorithm is the product of the expected number of stages and the expected reward in each stage. The lower bound for the total number of stages still holds (see (3.12)). Since the optimal policy for an l_0-depth SSAP problem is used by the Greedy SSAP-stage algorithm in each stage, then from Theorem 3.1, the expected reward in each stage is $\sum_{i=l_0-\min\{l_0,M\}+1}^{l_0} a_i^{l_0} w_{M-l_0+i}$. Combining the lower bound for the expected number of stages and the expected reward in each stage completes the proof. QED.

Theorem 3.3. The Greedy SSAP-stage algorithm is asymptotically $(2-\frac{1}{l_0})$-competitive for IID equal-length job sequences using multiple workers.

Proof. The result follows directly from Lemma 3.2 and Proposition 3.2. Let γ_{GSS} denote the competitive ratio of the Greedy SSAP-stage algorithm. Then,

$$\gamma_{GSS} = \frac{R_M^*}{\mathbb{E}[R_{GSS}]} \tag{3.38}$$

$$\leq \frac{(\lfloor (T-1)/l_0 \rfloor + 1) \sum_{i=l_0-\min\{M,l_0\}+1}^{l_0} a_i^{l_0} w_{M-l_0+i}}{\lfloor (T-1)/(2l_0-1) \rfloor \sum_{i=l_0-\min\{l_0,M\}+1}^{l_0} a_i^{l_0} w_{M-l_0+i}} \tag{3.39}$$

$$\leq \frac{\frac{T-1}{l_0}+1}{\frac{T-1}{2l_0-1}-1} \to 2-\frac{1}{l_0}, \tag{3.40}$$

as $T \to +\infty$. QED.

3.3 APPROXIMATION ALGORITHMS FOR RSSAP WITH A RANDOM ORDER OF ARRIVALS

This section maintains the assumption that all jobs have length l_0, but relaxes the assumption that job values are drawn from a given distribution, instead assuming that jobs have a random order of arrivals. Therefore, jobs are randomly ordered such that the i^{th}

arriving job is equally likely to have the j^{th} largest value for all $i, j = 1, 2, ..., T$, where T denotes the given total number of jobs. To analyze the algorithms in this setting, instead of using the approximation ratio from the previous section, we will analyze our algorithms using the more standard competitive ratio as defined in Definition 3.2.

Definition 3.2. Let OPT denote the optimal expected reward for an offline algorithm assigning job sequences in RSSAP. Let $R(\mathcal{A})$ denote the reward for assigning such job sequences using algorithm $\mathcal{A}$. Then algorithm $\mathcal{A}$ is said to be $\gamma_{\mathcal{A}}$-competitive where

$$\gamma_{\mathcal{A}} \equiv \frac{\mathbb{E}[\text{OPT}]}{\mathbb{E}[R(\mathcal{A})]}, \tag{3.41}$$

with the expectation in the numerator taken with respect to the random arrival order, and expectation in the denominator is taken with respect to both the random arrival order and the randomization of $\mathcal{A}$.

Our algorithm adapts the BOM algorithm, proposed by [35] for the weighted bipartite online matching problem. In the weighted bipartite online matching problem, right-side vertices R of an edge-weighted bipartite graph $G = (R \bigcup L, E)$ are given in advance. Left-side vertices arrive one at a time with their edges and weights of edges revealed. The BOM algorithm decides either to match an arriving left-side vertex to an unmatched right-side vertex or discard the left-side vertex upon each arrival: observes the first $\lfloor N/e \rfloor$ arriving vertices for training (N is the given total number of right-side vertices); beginning from the $(\lfloor N/e \rfloor + 1)^{th}$ arriving vertex, matches the new vertex based on the optimal (offline) matching for the set of vertices observed to date. [35] prove that the BOM algorithm is e-competitive for the weighted bipartite online matching problem. This is formally stated in Theorem 3.4.

Theorem 3.4 ([35])**.** Let R_{BOM} be the reward earned by the BOM algorithm on an instance of the online matching problem with n randomly ordered arrivals, and let OPT be the offline optimal reward on the same instance (note that OPT is deterministic, since the offline optimal solution does not depend on the arrival order). Then

$$\mathbb{E}[R_{\text{BOM}}] \geq \left(\frac{1}{e} - \frac{1}{n}\right) \text{OPT} \tag{3.42}$$

The BOM algorithm matches each vertex at most once, and hence, is not directly applicable to RSSAP.

This section considers equal-length job sequences. Let l_0 denote the length of each job and assume $l_0 \geq 3$. Let T denote the given total number of jobs. We consider a special

class of geometric arrivals with $p = 1$ (i.e., a job arrives at the beginning of each time slot with probability 1). Suppose that there are M available workers, with success rates $w_1 \leq w_2 \leq \ldots \leq w_M$.

We propose the *repeated BOM algorithm* for scheduling equal-length job sequences with a random order of arrivals. In particular, we apply the BOM algorithm to windows of l_0 jobs, then skip the next window to wait until all jobs are available again. We odd numbered windows with probability 1/2, and even numbered windows otherwise.

Algorithm 3.3 Repeated BOM algorithm

```
1: Let R = 1 with probability 1/2, let R = 0 otherwise
2: for the ith window (where i ranges from 1 to ⌊T/l_0⌋ and each window contains l_0 time
   slots) do
3:     if i ≡ R mod 2 then
4:         for the job J arriving in the tth time slot of this window do
5:             if t ≤ ⌊l_0/e⌋ then
6:                 discard J and continue
7:             else
8:                 compute the optimal matching M_{i,t} on all jobs seen so far in this window
9:                 let w be the worker J is assigned to in M_{i,t}
10:                if w has not been assigned during window i then
11:                    assign J to w
12:                end if
13:            end if
14:        end for
15:    end if
16: end for
```

Theorem 3.5 proves that the rolling window algorithm is $2\frac{l_0 e}{l_0 - e}$-competitive for equal-length jobs. Note that this is constant for fixed l_0, and approaches e as $l_0 \to \infty$. In particular, when $l_0 = 3$, we are guaranteed a 60-competitive algorithm, while with l_0 as low as 9, we are guaranteed an 8-competitive algorithm.

Theorem 3.5. The repeated BOM algorithm is asymptotically $2\frac{l_0 e}{l_0 - e}$-competitive for equal-length job sequences with a random order of arrivals.

Proof. We consider each window separately and take all expectations over the random arrival order and random choice of R. Let $\mathbb{E}[W_i]$ be the expected reward on window i, and recall that M_{i,l_0} is optimal offline matching on all of window i. Let $|M_{i,l_0}|$ be the value of matching M_{i,l_0} Let OPT be the value of the offline optimal solution, and let OPT_i be the value of the optimal reward earned by jobs in window i. Then for every i, we can write

$$\mathbb{E}[W_i] \geq \frac{1}{2}\left(\frac{1}{e} - \frac{1}{l_0}\right)\mathbb{E}[|M_{i,l_0}|] \geq \frac{1}{2}\left(\frac{1}{e} - \frac{1}{l_0}\right)\mathbb{E}[\text{OPT}_i], \tag{3.43}$$

where the first inequality follows from Theorem 3.4 and the fact that we use the BOM algorithm on a given window with probability 1/2, and the second inequality follows from the fact that M_{i,l_0} is the optimal matching on window i, and therefore is at least the value of OPT_i.

We can then sum over all windows. This gives

$$\mathbb{E}[R_{RBOM}] \geq \sum_{i=1}^{\lfloor T/l_0 \rfloor} \mathbb{E}[W_i] \tag{3.44}$$

$$\geq \sum_{i=1}^{\lfloor T/l_0 \rfloor} \frac{1}{2}\left(\frac{1}{e} - \frac{1}{l_0}\right)\mathbb{E}[\text{OPT}_i] \tag{3.45}$$

$$= \frac{1}{2}\left(\frac{1}{e} - \frac{1}{l_0}\right)\left(\mathbb{E}[\text{OPT}] - \mathbb{E}[\text{OPT}_{\text{last}}]\right), \tag{3.46}$$

where OPT_{last} is the portion of OPT attributed to any jobs arriving after the last full window of length l_0. Notice, however, that as $T \to \infty$, $\mathbb{E}[\text{OPT}_{\text{last}}]$ is negligible compared to $\mathbb{E}[\text{OPT}]$, and therefore asymptotically,

$$\mathbb{E}[R_{RBOM}] \geq \frac{1}{2}\left(\frac{1}{e} - \frac{1}{l_0}\right)\mathbb{E}[\text{OPT}], \tag{3.47}$$

which implies that

$$\frac{\mathbb{E}[\text{OPT}]}{\mathbb{E}[R_{RBOM}]} \leq 2\frac{l_0 e}{l_0 - e} \tag{3.48}$$

QED.

In RSSAP, the reward for assigning a job to a worker is given by the product of the worker's success rate and the job value. However, the rolling window algorithm can be generalized to cases where rewards for assigning a job J to a work w_m is given by any function $r(J, w_m)$, and Theorem 3.5 still holds.

3.4 CONCLUSION

This chapter considers RSSAP, a stochastic online matching problem with reusable resources, where workers are reusable and capable of performing more jobs after completing previously assigned jobs. Approximation algorithms are proposed for two kinds of fixed-

length job sequences: (1) jobs with IID values following a given distribution, and (2) jobs with a random order of arrivals.

RSSAP can also be classified as *stochastic online interval scheduling problems* for equal-length and memoryless-length job sequences. Job arrival times are discretized and values are drawn from a given distribution or a random permutation. Job assignments are assumed to be non-preemptive such that an assigned job has to be completed without interruption or termination. The reward for completing a job is the product of the job value and the machine weight. The approximation ratio of an online algorithm is the ratio of the optimal expected reward to the expected reward for the algorithm, with expectations taken over the distribution of the job sequence. The stochastic interval scheduling problem evaluates the average performance of online algorithms, which is different from the classic worst-case analysis for online interval scheduling problems. Results in this chapter show that meaningful competitive ratios can be derived with extra distributional information on the job sequence, in contrast to the infinite competitive ratio from worst-case analysis.

There are several directions to extend the work. Firstly, we only consider RSSAP on fixed length jobs. It would be interesting to extend these results to jobs with randomly generated lengths (e.g. from a geometric distribution). RSSAP with other classes of jobs is another possible direction to investigate. For example, another class of widely-studied jobs for online interval scheduling problems is C-benevolent jobs, whose values follow a non-negative, increasing, and convex function of lengths [36]. Finally, other classes of approximation algorithms may be explored. Approximation algorithms based on the Primal-Dual linear programs and the Cooperative Greedy algorithm [37] may be promising candidates. Last but not least, there is no lower bound for competitive ratios of approximation algorithms for RSSAP available in the literature yet. Deriving lower bounds for competitive ratios of online algorithms for RSSAP is another direction worth investigation.

REFERENCES

[1] S. R. Grenadier and A. M. Weiss, "Investment in technological innovations: An option pricing approach," *Journal of financial Economics*, vol. 44, no. 3, pp. 397–416, 1997.

[2] X. Su and S. A. Zenios, "Patient choice in kidney allocation: A sequential stochastic assignment model," *Operations Research*, vol. 53, no. 3, pp. 443–455, 2005.

[3] L. A. McLay, S. H. Jacobson, and A. G. Nikolaev, "A sequential stochastic passenger screening problem for aviation security," *IIE Transactions*, vol. 41, no. 6, pp. 575–591, 2009.

[4] S. H. Jacobson, G. Yu, and J. A. Jokela, "A double-risk monitoring and movement restriction policy for Ebola entry screening at airports in the United States," *Preventive medicine*, vol. 88, pp. 33–38, 2016.

[5] R. M. Karp, U. V. Vazirani, and V. V. Vazirani, "An optimal algorithm for on-line bipartite matching," in *Proceedings of the Twenty-second Annual ACM Symposium on Theory of Computing*, ser. STOC '90. New York, NY, USA: ACM, 1990. [Online]. Available: http://doi.acm.org/10.1145/100216.100262 pp. 352–358.

[6] M. Kao and S. R. Tate, "Online matching with blocked input," *Information Processing Letters*, vol. 38, no. 3, pp. 113 – 116, 1991. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0020019091902316

[7] J. Feldman, A. Mehta, V. Mirrokni, and S. Muthukrishnan, "Online stochastic matching: Beating 1-1/e," in *2009 50th Annual IEEE Symposium on Foundations of Computer Science*, Oct 2009, pp. 117–126.

[8] B. Bahmani and M. Kapralov, "Improved bounds for online stochastic matching," in *Algorithms – ESA 2010*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 170–181.

[9] V. H. Manshadi, S. O. Gharan, and A. Saberi, "Online stochastic matching: Online actions based on offline statistics," *Mathematics of Operations Research*, vol. 37, no. 4, pp. 559–573, 2012. [Online]. Available: https://doi.org/10.1287/moor.1120.0551

[10] B. Haeupler, V. S. Mirrokni, and M. Zadimoghaddam, "Online stochastic weighted matching: Improved approximation algorithms," in *Internet and Network Economics*. Springer Berlin Heidelberg, 2011, pp. 170–181.

[11] P. Jaillet and X. Lu, "Online stochastic matching: New algorithms with better bounds," *Mathematics of Operations Research*, vol. 39, no. 3, pp. 624–646, 2014. [Online]. Available: https://doi.org/10.1287/moor.2013.0621

[12] G. Aggarwal, G. Goel, C. Karande, and A. Mehta, "Online vertex-weighted bipartite matching and single-bid budgeted allocations," in *Proceedings of the Twenty-second Annual ACM-SIAM Symposium on Discrete Algorithms*, ser. SODA '11. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 2011. [Online]. Available: http://dl.acm.org/citation.cfm?id=2133036.2133131 pp. 1253–1264.

[13] A. Mehta, A. Saberi, U. Vazirani, and V. Vazirani, "Adwords and generalized online matching," *J. ACM*, vol. 54, no. 5, Oct. 2007. [Online]. Available: http://doi.acm.org/10.1145/1284320.1284321

[14] Z. Huang, N. Kang, Z. G. Tang, X. Wu, Y. Zhang, and X. Zhu, "How to match when all vertices arrive online," in *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*, ser. STOC 2018. New York, NY, USA: ACM, 2018. [Online]. Available: http://doi.acm.org/10.1145/3188745.3188858 pp. 17–29.

[15] Z. Huang, B. Peng, Z. G. Tang, R. Tao, X. Wu, and Y. Zhang, *Tight Competitive Ratios of Classic Matching Algorithms in the Fully Online Model*, 2019, pp. 2875–2886. [Online]. Available: https://epubs.siam.org/doi/abs/10.1137/1.9781611975482.178

[16] I. Ashlagi, M. Burq, C. Dutta, P. Jaillet, A. Saberi, and C. Sholley, "Edge weighted online windowed matching," in *Proceedings of the 2019 ACM Conference on Economics and Computation*, ser. EC '19. New York, NY, USA: ACM, 2019. [Online]. Available: http://doi.acm.org/10.1145/3328526.3329573 pp. 729–742.

[17] B. Gamlath, M. Kapralov, A. Maggiori, O. Svensson, and D. Wajc, "Online matching with general arrivals," 2019.

[18] A. Mehta, "Online matching and ad allocation," *Found. Trends Theor. Comput. Sci.*, vol. 8, no. 4, pp. 265–368, Oct. 2013. [Online]. Available: http://dx.doi.org/10.1561/0400000057

[19] C. Derman, G. Lieberman, and S. Ross, "A sequential stochastic assignment problem," *Management Science*, vol. 18, no. 7, pp. 349–355, 1972.

[20] S. Albright, "Optimal sequential assignments with random arrival times," *Management Science*, vol. 21, no. 1, pp. 60–67, 1974.

[21] A. G. Nikolaev and S. H. Jacobson, "Technical Note-Stochastic Sequential Decision-Making with a Random Number of Jobs," *Operations Research*, vol. 58, no. 4-part-1, pp. 1023–1027, 2010.

[22] S. Albright, "A Markov chain version of the secretary problem," *Naval Research Logistics Quarterly*, vol. 23, no. 1, pp. 151–159, 1976.

[23] T. Nakai, "A sequential stochastic assignment problem in a partially observable markov chain," *Mathematics of Operations Research*, vol. 11, no. 2, pp. 230–240, 1986.

[24] D. Kennedy, "Optimal sequential assignment," *Mathematics of Operations Research*, vol. 11, no. 4, pp. 619–626, 1986.

[25] D. T. Wu and S. M. Ross, "A stochastic assignment problem," *Naval Research Logistics (NRL)*, vol. 62, no. 1, pp. 23–31, 2015.

[26] G. Yu, S. H. Jacobson, and N. Kiyavash, "A Bi-criteria Multiple-choice Secretary Problem," *Technical Note Submitted*, 2016.

[27] A. G. Nikolaev, S. H. Jacobson, and L. A. McLay, "A sequential stochastic security system design problem for aviation security," *Transportation Science*, vol. 41, no. 2, pp. 182–194, 2007.

[28] L. A. McLay, S. H. Jacobson, and A. G. Nikolaev, "A sequential stochastic passenger screening problem for aviation security," *IIE Transactions*, vol. 41, no. 6, pp. 575–591, 2009.

[29] A. J. Lee, L. A. McLay, and S. H. Jacobson, "Designing aviation security passenger screening systems using nonlinear control," *SIAM Journal on Control and Optimization*, vol. 48, no. 4, pp. 2085–2105, 2009.

[30] A. G. Nikolaev, A. J. Lee, and S. H. Jacobson, "Optimal aviation security screening strategies with dynamic passenger risk updates," *IEEE Transactions on intelligent transportation systems*, vol. 13, no. 1, pp. 203–212, 2012.

[31] A. Tsopelakos and S. H. Jacobson, "Optimal policies for the sequential stochastic threshold assignment problem," University of Illinois, Urbana, IL, Tech. Rep., 2018.

[32] G. Yu, "Dynamic online resource allocation problems," Ph.D. dissertation, University of Illinois at Urbana-Champaign, 9 2018.

[33] D. P. Bertsekas, *Dynamic programming and optimal control*. Athena Scientific Belmont, MA, 1995, vol. 1, no. 2.

[34] G. H. Hardy and J. E. Littlewood, "A maximal theorem with function-theoretic applications," *Acta Mathematica*, vol. 54, no. 1, pp. 81–116, 1930.

[35] T. Kesselheim, K. Radke, A. Tonnis, and B. Vocking, "An optimal online algorithm for weighted bipartite matching and extensions to combinatorial auctions," *Algorithms-ESA*, 2013.

[36] G. J. Woeginger, "On-line scheduling of jobs with fixed start and end times," *Theoretical Computer Science*, vol. 130, no. 1, pp. 5–16, 1994.

[37] G. Yu and S. H. Jacobson, "Online C-benevolent Job Scheduling on Multiple Machines," *Optimization Letters*, pp. 1–13, 2017.